QoS-Based Web Services Composition using Water Flow-like Algorithm

Mahyar Taheri[1], Babak Shirazi[2], Hamed Fazlollahtabar[3]

[1]Department of Information Technology Engineering, Mazandaran University of Science and Technology Babol, Iran

[2]Department of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran

[3]Faculty of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran

[1]mahyar.tahery@gmail.com; [2]shirazi_b@yahoo.com; [*3]hfazl@iust.ac.ir

Received Mar 26, 2014; Accepted Apr 21, 2014; Published Jun 12, 2014

Abstract

Web services composition is one of the most important issues in service oriented architectures. For responding to the majority of complicated business process, it might not have been just a single service, so several services must have been combined to reach a suitable one. Composite service will be generated by combining single web services. Each web service may have different implementations with similar functions, but something making it different from other similar services is the quality of service (QoS). In this study, QoS-based web service composition is considered and an architecture for automated web service composition is proposed. In this architecture, at first users enter their functional and non-functional requirements into the system by user interface, then a water flow-like algorithm (WFA) is developed for optimal composition of web services in order to indulge users requirements in a suitable time. Proposed approach is implemented and evaluated by C# language. The evaluation results have shown that the WFA outperforms Improved GA and simple GA when the number of abstract services is large.

Keywords

Web Service Composition; Quality of Service; Water Flow-like Algorithm

Introduction

Today, because of increase in flow of information inside and outside of organization and its management, organizations don't have any choice except using the advantages of Information Technology and information systems. Service Oriented Architecture offers new model in implementation of informational system and lets the system developers focus most of their attentions and realizing characters which the organization needs (Griffiths and Chao, 2010). There are various definitions for Service Oriented Architecture which is often based on the usage of it. (MacKenzie, 2006) A definition of Service Oriented Architecture is given by OASIS: It is a model for organizations and the usage of their distributed abilities is under control by different people. Based on the given definition , Service Oriented Architecture is a model. In other word , SOA is an approach or a meditative method, so we can't say that is a tool, which we can buy (Josuttis, 2007). Most advantages of using SOA include: Reusability of services, its ability in decreasing the expenses, improvement organization's agility and business (Svensson and Wallen, 2006). The other strong point of it is in leveling the operations between heterogeneous information systems. It will be used for integration and connection between informational systems of web services, but in most cases a single service can't offer all complicated requirements of customers alone, because of that , answering to these requirements should be used by multiple services. So the ability of the organizations and companies in selection and composition of web services in developing software and informational services will become very important in order to use the functions and variety of services could offer the customers' requirements and their complex requirements (Ma and He, 2009).

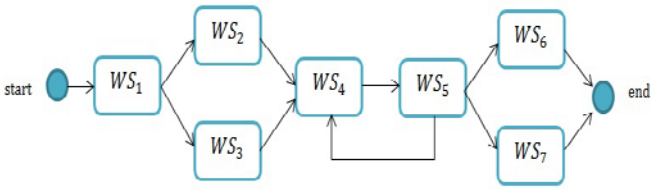

FIG. 1 SERVICE COMPOSITION IN WORKFLOW

When an organization requests service composition, at first it should design a workflow of activities that each

of them performs with a Web service. Fig. 1 shows workflows of organization's activities that include 7 web services (WS_1, WS_2, WS_3…, WS_7). Any of these web services can have different implementation but similar function. Users who order web services usually express their non-functional requirement with quality criterion. The quality of service defines the abilities of a product or a service for facing with non-functional requirements of a user and these criteria can be used as a benchmark for differentiate and distinction between services and the service provider (Sha, 2009). There might be services between the similar ones that are based on quality criteria which will be more efficient for users, therefore there is a period of time which is given to perform activities existed several services with similar functions, then the service would be chosen for a user based on requirements and quality of a service. In this study we consider 5 qualities of services which are: Response time, latency, reliability, availability and success rate which are more popular than others however the proposed approach will develop in a way that the new quality criteria can be added easily or omit one of them. The related paramets and their descriptions are given in Table 1.

TABLE 1 QWS PARAMETERS AND UNITS

Parameter Name	Description	Units
Response Time	Time taken to send a request and receive a response	ms
Availability	Number of successful invocations/total invocations	%
Reliability	Ratio of the number of error messages to total messages	%
Latency	Time taken for the server to process a given request	ms
success rate	Number of response / number of request messages	%
Response Time	Time taken to send a request and receive a response	ms

Discovering and composition of services on the run time of that also are the most important part of progressive challenge in service oriented application.

By increasing the use of service-oriented technologies and web services and considering the competitive rival amongst the web service providers, several providers evolved having similar performance but with different quality. Services can be combined in several ways, thus obtaining an optimal combination by searching all cases is complicated.

If we have all number of activities in workflow (k) and for performing each of them , there is a web service with a similar function (n), in this situation n^k causes different composition of web services for performing work processes. Therefore achievement of an optimal composition from different existed compositions will be very complicated and searching all cases is very expensive and time consuming. On the other hand , the number of produced composition by considering the quality criteria of services will increase (Wang, 2007; Zhang, 2011). Thus, because of the intensive computation, in this study , we use the Water Flow-like Algorithm for optimization of selection and composition of web services, which has the feature of multiple and dynamic numbers of solution agents, and its mimicking of the natural behavior of water flowing from higher to lower levels coincides exactly with the process of searching for optimal solutions (Yang and Wang, 2007). In this model for determining , whether the produced composition of web services is a good one or not a kind of fitness function is used. With respect to the way solutions generated in WFA, the complexity of the algorithm to combine web services is nlogn, having less complexity comparing to GA with n2.

The remainder of the paper is organized as follows. After a review of the literature of web services selection in Section 2, in Section 3 we explain the way how to make formula and un-scaling the quality of services. In the Section 4, the way to operate the fitness function for composing web service is given. In Section 5, we present the proposed architecture and in the Section 6 the approach will be evaluated and finally conclusion and the future work are presented in Section 7.

Related Work

These days' composition and selection of web services change into a very important subject in academic issues based on quality of service. The approach of selection and ranking web services is provided by Yin and You's quality of services. In this approach to supervise services' QoS and facilitate the service evaluation, we assume that all users' concerned services' QoS aspects are controlled and reported to the service directory (SD), which saves services' QoS information along with their registrations. This information will be applied by the service evaluator to rank services and help users select services when there are more than one registered services satisfying users' functionality requirements (Yau and Yin, 2011).

The framework of DynamicCos is an automatic mechanism for composition of services in execution

time. This architecture supports all the phases and stakeholders of the dynamic service composition life-cycle, as automatic service discovery, selection and composition and with the CLM, the composition algorithm has to find a composition of services that fulfill the service request (Goncalves da Silva et al. 2009).

SODIUM (Service-Oriented Development in a Unified fraMework) is an international project, involving research, technological and industrial partners, dedicated to tackling interoperability challenges that companies face at the data, services and business levels. The project has developed a Generic Service Model, containing the common concepts of heterogeneous services from multiple points of view. The special characteristics of individual service technologies (such as Web services, Grid services or P2P services) are then dealt with as extensions to the core. The SODIUM methodology adopts a model-driven and iterative approach for service composition. The approach utilized OMG's QoS profile to represent collections of QoS properties (Topouzidou, 2007).

One of approaches of selecting web services based on quality criteria takes place by using a 2- dimensional Boolean array which is called selection matrix. Row of matrix represents web services and the columns show quality of services. When the value of matrix's cell equals 1, it means that the user's requirements of quality of service are the same as published value for service and otherwise the value equal 0. Then when the matrix row has maximum number of 1 we can conclude that service can be the best web service for meeting the user's quality of service requirements (Chaari et al., 2008).

Multi Objective Genetic Algorithm (MOGA) is presented for optimal selection of web service based on quality of services. In this algorithm , at first a mechanism is designed for possible combinations and feasible web services according to constraints of services composition (like dependency constrains and conflict between web services) and more genetic operators (selection operator, cross over and mutation) and strategies for distribution of diverse population that these populations are the same various composition of web services is introduced (Wang and Hou, 2008).

A flexible approach is presented by Yang and Chun Hung for composing services based on genetic algorithm which can do the composition of web services based on users' functional and non-functional requirements. In the presented approach for composition of workflows, like sequential, conditional, fork, is considered. Procedures in the presented approach is as follows that the users send their requirements in the algorithm and then the algorithm of the best composition for meeting the requirements of the users among existing services in UDDI is provided (Fanjiang et al., 2010).

A flexible architecture is presented by Bounhamdi and Jarir for dynamic web service composition at run time. In this firstly a general framework for ranking and selection of services with characteristics of presented WSSR-Q which is based on service description model. Then an algorithm will be presented for ranking of service in order to calculation and normalize service quality values for all web services. Afterwards a quality updating mechanism will be presented to update the quality which is based on users' feedback values (Zou et al., 2009).

Another approach to select and compose efficiently of web services is using the Particle Swarm Optimization which has been presented by Zhongjun Liang. In this composition approach , some constraint for choosing web services is presented. Constraints of the qualitative requirements of laws are designed. These constraints are intended between services on the implementation approach. In this approach at first a list of impossible composition is mentioned and then Particle Swarm Optimization algorithm is used among feasible combinations (Liang et al., 2012).

Selection of a suitable service according to functional and non-functional requirements of users is one of the most important goals of selection and composition of web services, some approaches, which suggest the selection of web services based on quality of service, have been addressed. Then we will mention some disadvantages and shortcomings of available algorithm. In some presented approaches , optimal selection of web service are considered dependent from others and there is no constraint for combining services, so in these ways when candidates service for selecting, dependent on the other services, can't have good selection. Some presented approaches of Quality of Service matrix (QoS matrix) are using qualitative features for calculating. The basis of some of presented methods of matrix are on computing properties of quality of service and while the number of service quality or web service increases, the matrix becomes very big and the number of calculations increases and becomes complicated so using these

methods is not always suitable (Tao and Yang, 2008; Rosenberg, et al., 2008).

Some of existing methods don't support multiple quality criteria of users. In these approaches , the user can just one quality of service for selecting a web service. Also in some other available approaches , although they support different quality of service but the user can't prioritize for each of quality of service by weighting (Paulraj et al., 2012).

But in this study , an approach is presented for the automatic and dynamic composition of Web services, this approach uses a meta-heuristic algorithm in order to increase the scale of problem, which gives a suitable composition to users in a reasonable time and also can prioritize each quality by weighting of each quality of services which were considered before. The details are given later.

Modeling

In this study, when we talk about an abstract web service , we refer to a web service in a workflow and while we say concrete web service , it is the implementation of an abstract web service.

WS_i: i is an abstract web service in a workflow.

$WS = (WS_1, WS_2, \dots WS_i, .., WS_n)$

On top $(1 \le i \le n)$

$S_{i,j}$: A concrete web service of j is in an abstract web service i.

On top $(1 \le i \le n)$ and $(1 \le j \le K_i)$ and also K_i is the total number of concrete web services for each of the WS_i abstract web services.

$q_{i,j}^l$: The L[th] QoS for concrete web services of j which on the i[th] abstract service

Q= $(q_{i,j}^1, q_{i,j}^2, q_{i,j}^3, .. q_{i,j}^m)$

where m is total number of quality service and $(1 \le i \le n)$ and $(1 \le j \le K_i)$ and $(1 \le l \le m)$.

C^l: Each of the criteria for quality of service, to assign a weight due to its importance for the user.

C= $(c^1, c^2, c^3, c^4, \dots, c^m)$

where m is the total number of quality of service and $(1 \le l \le m)$ also sum of total QoS criteria is equal to 1 ($c^1 + c^2 + c^3 + c^4 + \dots + c^m = 1$).

Each of the presented quality of service in this research has different feature and sometimes they probably conflict each other, therefore in order to compare different Criteria, at first we should un-scale this QoS by a mechanism. In this un-scaling method, for comparable value of these data of QoS to each other, by at least (q_{min}^l) and maximum (q_{max}^l) of these which are found in services by research, we transfer all the data of quality of service in 0 and 1. We use equation 1 for the QoS that has direct relationship with the criteria of quality and equation 2 for the QoS which has vice versa relationship.

$$Q_{i,j}^l = \begin{cases} \frac{q_{i,j}^l - q_{min}^l}{q_{max}^l - q_{min}^l} & if(q_{max}^l != q_{min}^l) \\ 1 & if(q_{max}^l = q_{min}^l) \end{cases} \quad \text{Equation1}$$

$$Q_{i,j}^l = \begin{cases} \frac{q_{max}^l - q_{i,j}^l}{q_{max}^l - q_{min}^l} & if(q_{max}^l != q_{min}^l) \\ 1 & if(q_{max}^l = q_{min}^l) \end{cases} \quad \text{Equation2}$$

Fitness Function

Water flow-like algorithm uses a fitness function to evaluate each element of the current solution. This function computes fitness of a composition services with different structures, according to Quality of Service and user weighting. Each of the single services can be combined with various structures including: sequential, fork, conditional and loop. In this study , the dynamic service composition is provided and according to requirement of user, single services can be combined to each other. Therefore calculated fitness of a composite service, according to the structures of used is different and after analyzing the process model, it is calculated by the specified user.

TABLE 2 AGGREGATION METHOD OF QOS PROPERTIES

	sequential	fork	conditional	loop
response time	$\sum_{i=1}^{n} T(S_i)$	$Max(T(S_1) \dots T(S_n))$	$Max(T(S_1) \dots T(S_n))$	$k * T(S)$
Latency	$\sum_{i=1}^{n} L(S_i)$	$Max(L(S_1) \dots l(S_n))$	$Max(L(S_1) \dots L(S_n))$	$k * L(S)$
Availability	$\prod_{i=1}^{n} Av(S_i)$	$\prod_{i=1}^{n} Av(S_i)$	$Min(Av(S_1) \dots Av(S_n))$	$(Av(S))^k$
Reliability	$\prod_{i=1}^{n} Re(S_i)$	$\prod_{i=1}^{n} Re(S_i)$	$Min(Re(S_1) \dots Re(S_n))$	$(Re(S))^k$
success rate	$\prod_{i=1}^{n} Se(S_i)$	$\prod_{i=1}^{n} Se(S_i)$	$Min(Se(S_1) \dots Se(S_n))$	$(Se(S))^k$

In Table2 there is a list of quality criteria used in this research and along with the way of computing,

aggregating the QoS property values in different structures is mentioned, in order to compute fitness function, one complicated Service at first step, the value of each quality of service criteria is in Table2 according to the different structures and therefore aggregating the QoS property values in complicated service is computed. At 2nd step, aggregating the QoS property values is mentioned in the previous step to un-scaling and after doing that, weights which are determined by user are added to each till the fitness function for a complicated service is computed.

As it is shown in Fig. 2 in computing the aggregating of the QoS property values, services which are in connection to each other , similar structures are determined as virtual service and it can be connected by the other services. As an example at Fig. 2, two services S_2 and S_3 are combined to each other by sequential structure and virtual service (Sequential (S_2, S_3)) are made and this virtual service is as a fork with service S_4 and create virtual service (Fork (Sequential (S_2,S_3),S_4)). And the structures and the way to connect to other abstract service will be determined (Ko et al., 2008). As the composition of service which is presented on the approach is dynamic and each user requests his specific composition, therefore in order to compute fitness function, at first structural model should be determined for different QoS criteria.

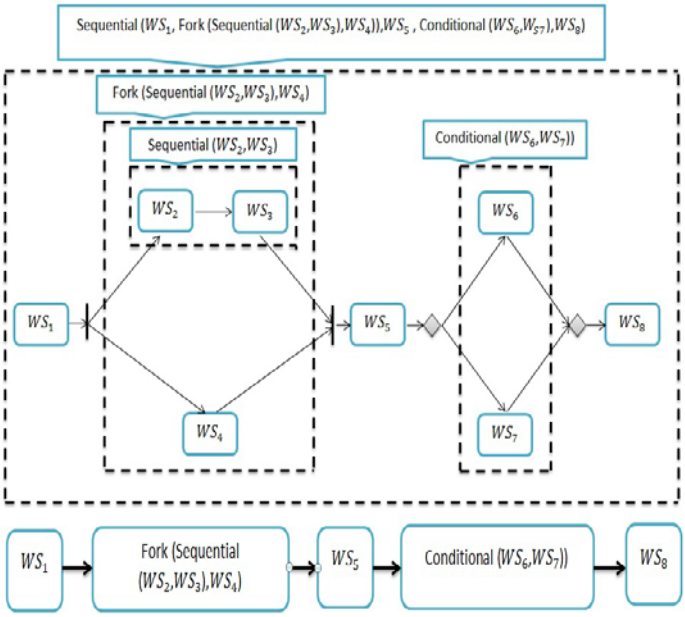

FIG. 2 AGGREGATION PROCESS EXAMPLE OF QOS PROPERTIES

For the process model of above mentioned, after analysis on the model, as it is clear at Fig. 2, this structural model for each of QoS criteria (q^l) is acquired of equation 3.

SM_{q^l} =Sequential($q^l(S_1$), Fork(Sequential ($q^l(S_2)$, $q^l(S_3)$) , $q^l(S_4)$), $q^l(S_5)$, Conditional($q^l(S_6)$,$q^l(S_7)$),$q^l(S_8)$) Equation3

By replacing the value of QoS in the acquired structural model, the value of criteria is computing in composition of services. It continue in order to computing fitness function of a complicated service, it needs to use the equations 1 and 2 which were explained before and for q^l which have inverse relation by quality criteria, such as response time and latency, SM_{q^l} is in equation 2 and for the others q^l, SM_{q^l} is in equation 1. Therefore fitness function of the made complicated service at Fig. 2 is acquired by this formula.

Equation4

$$F = \sum_{l=1}^{2} C^l * \left(\frac{SM_{q^l_{max}} - SM_{q^l}}{SM_{q^l_{max}} - SM_{q^l_{min}}} \right) + \sum_{l=3}^{5} C^l * \left(\frac{SM_{q^l} - SM_{q^l_{min}}}{SM_{q^l_{max}} - SM_{q^l_{min}}} \right)$$

Proposed Approach

In this section, we introduce proposed approach and the architecture. Our proposed architecture for efficient Web services composition is shown in Fig. 3. Afterward every one of these parts is explained separately.

User interface: in this architecture , the functional and non-functional requirements of requester are received by a user interface, and changed into a process model by Business Process Execution Language.

Candidate service selection: when users request a complicated service instead of searching all services in the repository, at first, a number of services that meet the functional requirements of the service requester are selected as the candidate service

QoS DataBase: it's a Data base in which the values of QoS criteria of all existing web services in the service repository are stored.

Service repository: in this part, there are different web services to support the different requirements of users and after that new service is registered in UDDI by providers, repository service will be updated and web

services' quality will be held in service repository.

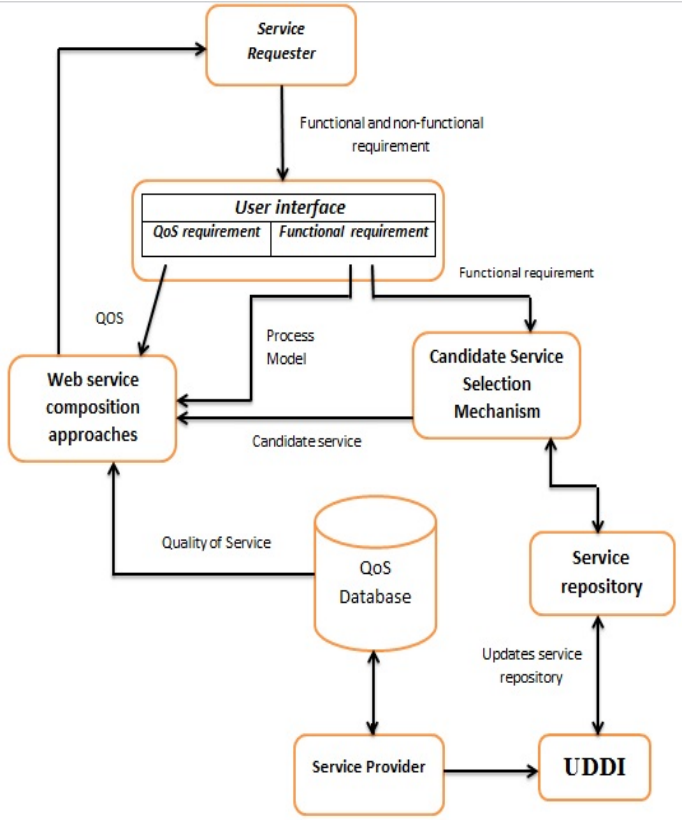

FIG. 3 SUGGESTED ARCHITECTURE FOR QOS-BASED WEB SERVICE COMPOSITION

Web service composition approach: in this step, all candidate service and quality requirements of users which were achieved before, go into service composition approach. Now by using one Meta heuristic approach we search among all candidate service according to the QoS criteria till select the best composition between different situations. Finally after finding the most suitable composition, according to the requirement of requester, this selected service is presented to the user. Our proposed approach is based on water flow in nature. In this approach, we use the water flow-like algorithm (WFA) which is inspired by natural behavior of water flowing from a higher level to a lower level.

Water Flow-like Algorithm:

This paper adopts the WFA logic and designs a heuristic algorithm for solving the web service composition problem because the WFA has the features of multiple and dynamic numbers of solution agents. The water flow-like algorithm (WFA) is inspired by natural behavior of water flowing from a higher level to a lower level. In this paper solution agents are modeled as water flows and the fitness function surface on the solution space is modeled as the terrain traversed by the flows. WFA has the features of multiple and dynamic numbers of solution agents, in fact Yang and Wang (2007), proposed the WFA to overcome the shortcomings of single- and multiple-solution- agent-based algorithms.

In the beginning, only one water flow is defined at a randomly generated location. In addition, the initial mass and velocity are given. Afterwards, the flow splits into multiple sub-flows when rugged or broken terrains encounter. Therefore, the number of flows (agents) is changing in quest for the optimal solutions. The movements of the flows can be controlled by the gravitation force and the energy conservation law (fluid momentum); flows can run to higher attitudes, once the kinetic energy is larger than the required potential energy, to navigate various terrains and avoid being trapped within a local minimum. The altitude of a flow represents the fitness function value. Water moving to a lower position can be considered as a solution searching for the optima (Wu et al., 2010).

The design ideas of adapting the natural behavior of water flow (Yang and Wang, 2007), are summarized as follows:

1. Driven by the gravitation force and governed by the energy conservation law, water will constantly flow to lower attitudes. Therefore, the solution search will recursively move to superior solutions from inferior ones iteration by iteration.

2. Fluid momentum drives water moving forward against rough terrains. A flow will split into sub-flows when it encounters rugged terrains and its momentum exceeds an amount for splitting. WFA simulates this behavior as an agent forking operation; i.e., more than two agents are derived from a single agent. A flow with larger momentum will generate more streams of sub-flows than a lower one. A flow with limited momentum will yield to the landform and maintain a single flow. Therefore, the fluid momentum of a flow is recalculated for determining the number of sub-flows that can be forked after each move.

3. Water flows to lower attitudes and occasionally springs up or swells up to higher attitudes, once the kinetic energy is larger than the required potential energy. To avoid being trapped within a local minimum, WFA allows the water to flow to a worse location to broaden the exploration area.

4. A number of flows merge into a single flow when they run to the same location. WFA reduces the number of solution agents when multiple agents move

to the same location, to avoid redundant searches.

5. Water flows are subject to water evaporation in the atmosphere. The evaporated water will return to the ground by rainfalls. In WFA, a part of the water flow is manually removed to mimic the water evaporation. A precipitation operation is implemented in WFA to simulate natural rainfall to explore a wider area.

General statement of WFA is presented in Fig. 4. Also, the operators are described below:

Split_Move: An operator to move and split water flow that a more intensive flow makes more subflow.

Flow_Merge: An operator to merge flows. Flows having equal fitness functions are merged.

Evaporation: Flows that no improvement considered for them are evaporated to fall again using rain operator in another space.

Rain: Using this operator an evaporated flow is fallen in a space which was not explored before.

```
Set_Parameter();
While (timePeriod not expired) do
        Split_Move();
        Flow_Merge();
        Evaporation();
        if Rainfall_Required=true then
                Rain();
                Flow_Merge();
        end // if
        if New_Best_Solution=true then
                Update_Best_Pool();
        end // if
End // While
```

FIG. 4 A PSEUDO CODE FOR WFA

Operations of WFA:

Here we adopt the WFA operations considered in (Yang and Wang, 2007), for our problem of QoS-Based Web Services Composition utilizes the feature of multiple and dynamic numbers of solution agent on maximizing aggregate the QoS property values. In this research, each of the flow represents a web service selection plan and it is encoded in an array of n integers $x_1, x_2, \ldots, x_n$, where n is the total number of abstract web services in the workflow of the composite web service.

1) Movement and splitting operator

At the beginning of the WFA, it is assumed that there is only one water flow at a randomly generated location. Yang and Wang (2007) and Wu et al. (2010) define a momentum base moving and splitting operation, by defining velocity and mass for each flow, but we define a mechanism in which velocity of each flow is based on rate of improvement in the solution. This means that a sub-flow with more improvement, gains more momentum and flows with higher momentum, generates more sub-flows than a lower one

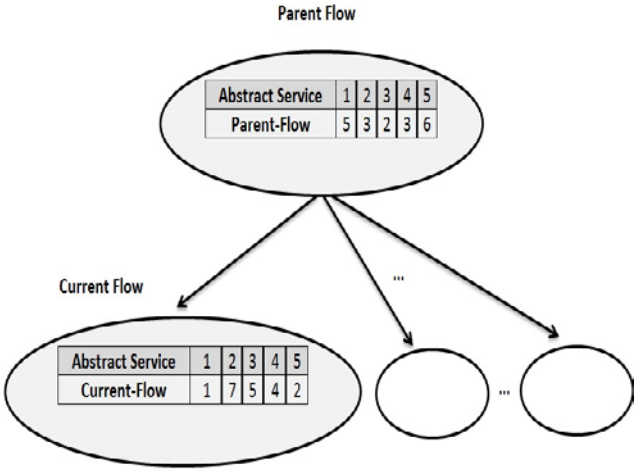

FIG. 5 WFA ENCODING SCHEME

When a flow does not split, it goes on as a single stream to the best feasible neighboring location. Before proceeding to the descriptions, we introduce some notation:

n_i: Number of sub-flows forked from flow i.

x_i: Solution corresponding to flow i.

x_{ik}: Solution corresponding to sub-flow k split from flow i.

$\bar{n}$: Upper limit on the number of sub-flows split from a flow.

It is possible that the number of sub-flows grows exponentially and exhausts the computational resource. Yang and Wang (2007) suggests imposing an upper limit, $\bar{n}$, on the number of sub-flows forked from a flow at each iteration. The number of sub-flows split from a flow can thus be obtained through the formula below:

$$n_i = min\ \{max\{int\ (f(x_i) - f(x_{ik}) * 300\ ), 0\}, \bar{n}\}$$

When number of sub-flows are forked from a flow equals to zero (no improvement on sub-flows), the momentum delivered to sub-flows has been used up, implying that these sub-flows will stagnate in its current location (solutions trapped in local optima)

with no splitting and movement. Such stagnant flow will gradually evaporate into the atmosphere, returning to the ground by precipitation later on. Pseudo code of this movement and splitting operator are shown in Fig. 6.

```
Function
Movment_Splitting_Flow(Parent_flow,Current_flo
w,p)
  for  i = 0 to Current_flow.Length() Do
    R = Random(0,1);
    if R < p then
     Sub_flow[i] = Current_flow [i];
    End //if
    else
          Sub_flow[i] = Parent_flow[i];
    End //else
  End // for
  return  Sub_flow;
End //Function
```

FIG. 6 MOVEMENT AND SPLITTING FLOW IN WATER FLOW LIKE ALGORITHM

In above, p is a parameter which is determined by a user and in this research equaled to 0.7

2) Flow-merging operation

When more than two flows move to the same location, they will merge into one flow with a bigger mass and momentum Using the flow-merging operation, the WFA reduces the number of solution agents when multiple agents result in the same objective value to avoid redundant searches. Pseudo code of this Flow merging operator is shown in Fig. 7.

```
Function Flow_Merge (Current_flow,FlowList)
 for i=0 to FlowList.Length() Do
  if Current_flow.Fitness == FlowList[i].Fitness then
    FlowList[i].Remove();
  End //if
 End//for
```

FIG. 7 FLOW MERGING IN WATER FLOW LIKE ALGORITHM

In above, FlowList is a list of all flows which is in the same level with current flow.

In this function, current flow is compared with all flows of its level and if its fitness function be equal to flows list, these flows will be deleted till stop extra search.

3) Water evaporation and precipitation operation

To simulate the natural evaporation of water into the air, a water evaporation operation is executed in WFA. After a prescribed number of iterations, the evaporated water will return to the ground by precipitation. The returned locations are stochastically determined from the locations of the current flows. Water evaporation and precipitation coincide with the 'escaping from local optima' mechanism that was used to avoid being trapped and to explore more solution spaces. After the splitting operation, if there is not any improvement on sub-flows, we will cease (stagnate the water) and force flows to evaporate into the atmosphere and come back to another space of that subject as precipitation. In order to implementation of this operator, when a new flow is going to split, it compares with its parent to determine, any improvement has been achieved or not. Pseudo code of Water evaporation and precipitation operation are shown in Figures 8 and 9.

```
Function Evaporation (Current_flow)
        R = Random(0,n);
         B=DecimalToBinary(Current_flow[R])
                X=ChangeBit(B);
                D=BinaryToDecimal(X);
                Current_flow[R]=D;
        Return Current_flow;
End //Function
```

FIG. 8 WATER EVAPORATION IN WATER FLOW LIKE ALGORITHM

```
Function precipitation (Parent_flow, Current_flow)
   if Current_flow.Fitness <= Parent_flow.Fitness then
                Evaporation(Current_flow);
End //Function
```

FIG. 9 PRECIPITATION IN WATER FLOW LIKE ALGORITHM

4) Searching neighborhood solutions

Among the four operations in the WFA, the flow splitting and moving operation mission is to search for better neighborhood solutions and ultimately, select the best solution of the current solution. By using this operation, we can split our randomly generated solution iteratively into sub-flows, traverse the solution space, and move toward the optimal solution. In the water flow-like algorithm, the locations of the split sub-flows are derived from the neighboring locations of the original flow with a little change on value, flows are split. For example, flow x_i which includes sub-flows x_1 , x_2,… x_k by searching all of

sub-flow x_i, the best neighbor can be found around it and it can be split for sure to have the best neighbors, we use a parameter names the number of neighbors search. For example if the parameter of searching neighbors is equal to 3, we should change one pair of a fraction part of a flow three times that shows the concrete web services assigned on each abstract web service. Finally the neighbor which has the best fitness will be chosen as the best neighbor for the flow. In Fig. 9 split and movement operator for searching the neighborhood of the flow web service composition problem is showed.

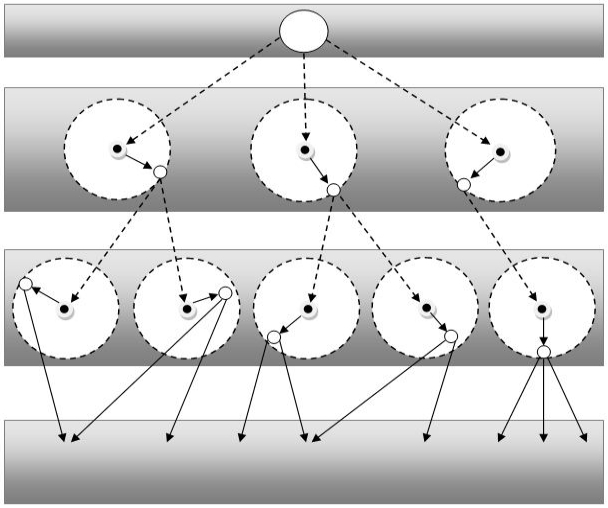

FIG. 10 FLOW SPLITTING AND MOVING OPERATION FOR SEARCHING NEIGHBORHOOD SOLUTIONS

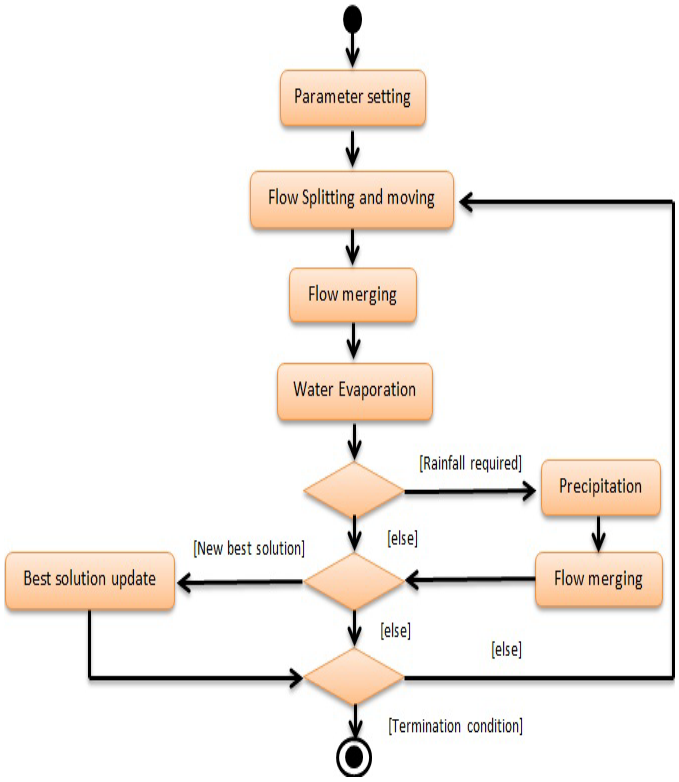

FIG. 4 COMPUTATION FLOW OF WFA [18]

Evaluation

The computation time and quality of the results produced by the WFA and GAs depend on the size and the complexity of the web service selection problem. The size of the problem is dependent on two parameters. 1) The number of abstract web services in the workflow. 2) The number of concrete web services for each of the abstract web services.

The first set of test problems included 10 test problems with different numbers of abstract web services. The number of abstract web services ranged from 5 to 50 with an increment of 5. The number of concrete web services for each of the abstract web services was fixed to 20.

The second set of test problems also included 5 test problems. The number of concrete web services for each of the abstract web services ranged from 10 to 50 with an increment of 10. The number of abstract web services was fixed to 10.

In all test cases, service composition is generated randomly by different structure including sequential, fork, conditional and loop. Also values of QoS criteria for abstract web service use a real Dataset (Al-Masri and Mahmoud, 2007; , Al-Masri and Mahmoud, 2007). QWS is a real dataset and represents 2500 real web services that exist on the Web. In this paper in order to test the performance of WFA, IGA and simple GA, they were implemented in Microsoft Visual C# 2008 and all the experiments were conducted in a PC with a 2.2 GHz Intel Core 2 Duo CPU and a 2 GB RAM.

Firstly, WFA, IGA and simple GA were used to solve each of the test problems in the first test set. Considering the stochastic nature of the WFA and GAs, each of the experiments was repeated 10 times and then calculated the average fitness value and average execution time for each of the experiments for each of the WFA and GAs.

Table 3 shows the fitness of WFA, IGA and simple GA for first test set and it includes 10 test case and Fig. 12 shows that by increasing these abstract services from 5 to 50, what kind of changes will happen in the execution time. As it was clear from Table 3, WFA in 10 test cases had better performance and had better fitness than the IGA and simple GA. Also according to Fig. 12, Water Flow-like Algorithm by increasing the number of abstract services from 5 to 50 in all test cases has less execution time than IGA and simple GA. Therefore above results show better performance of WFA than IGA and simple GA in terms of fitness and

execution time. In continuing WFA, IGA and simple GA were used to solve each of the test problems in the second test set. Considering the stochastic nature of the WFA and GAs, each of the experiments was repeated 10 times and then calculated the average fitness value and average computation time for each of the experiments for each of the WFA and GAs. Table 4 shows average fitness of WFA, IGA and simple GA for second test set and Fig. 13, shows increase of average execution time for WFA, IGA and simple GA when the number of concrete web services ranged from 10 to 50.

TABLE 3 FITNESS OF WFA, IGA AND SIMPLE GA BY INCREASING THE NUMBER OF ABSTRACT SERVICES

Abstract Service	WFA	IGA	SGA	Percent
5	**0.33373**	0.33304	0.32273	0.20718232
10	**0.33293**	0.33242	0.32219	0.153420372
15	**0.32936**	0.32896	0.32832	0.121595331
20	**0.32941**	0.32907	0.32841	0.103321482
25	**0.32949**	0.32849	0.32789	0.30442327
30	**0.32922**	0.32898	0.32717	0.072952763
35	**0.3288**	0.32615	0.32521	0.812509581
40	**0.32773**	0.32663	0.32523	0.336772495
45	**0.32715**	0.32648	0.32473	0.205219309
50	**0.32708**	0.32696	0.32458	0.036701737

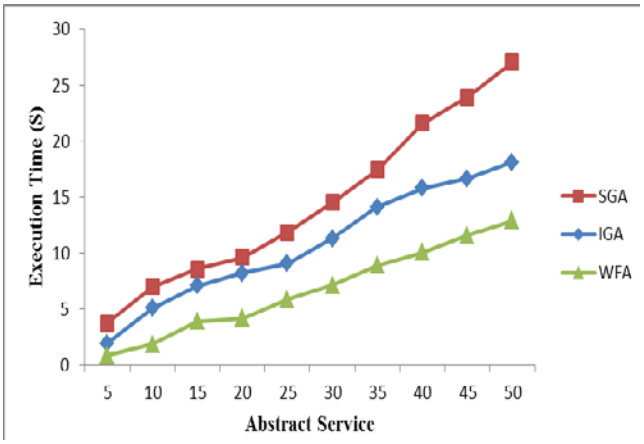

FIG. 5 EXECUTION TIME OF WFA, IGA AND SIMPLE GA BY INCREASING THE NUMBER OF ABSTRACT SERVICES

TABLE 4 FITNESS OF WFA, IGA AND SIMPLE GA BY INCREASING THE NUMBER OF CONCRETE SERVICES

Abstract Service	WFA	IGA	SGA	Percent
10	**0.33365**	0.33235	0.32225	0.391153904
20	**0.33348**	0.33237	0.32228	0.333965159
30	**0.33336**	0.33218	0.32216	0.355229093
40	**0.33303**	0.33219	0.32203	0.252867335
50	**0.33284**	0.33212	0.32211	0.216789112

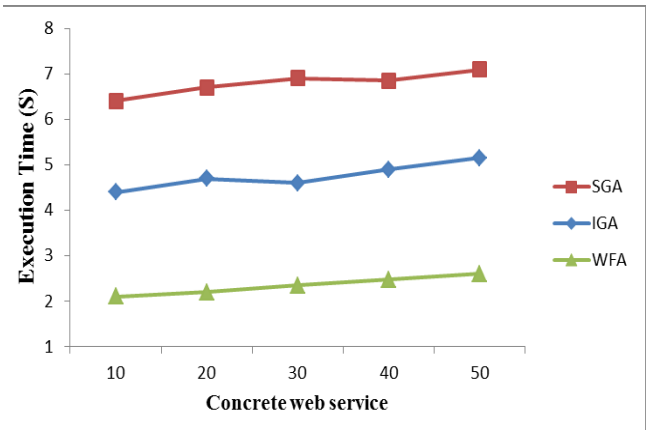

FIG. 6 EXECUTION TIME OF WFA, IGA AND SIMPLE GA BY INCREASING THE NUMBER OF ABSTRACT SERVICES

According to Table 4 in terms of the fitness by increasing the number of concrete services , WFA has better performance than the IGA and simple GA in most test cases and as it was clear, in the Fig. 13, WFA, IGA and simple GA had no correlation with the number of concrete web services per abstract web service. WFA has the execution time less than 3 seconds for all test cases and it has better execution time than IGA and simple GA.

Scalability: All the meta-heuristics are scalable so that the computation time of them increases linearly when the number of abstract web services increases. It does not change significantly, especially for WFA, when the number of concrete web services increases or when the number of constraints increases. As the slowest algorithm of the three GAs, the SGA takes less than 30 seconds for all the test problems, which is acceptable for design time web service-selection problems. In addition, among the provided approaches, the WFA is the fastest.

Performance: All the three meta-heuristics are effective. They can find the optimal solutions or the suboptimal solutions for most complex problems, where the constraint densities are high. The WFA outperforms the two other approaches in terms of solution quality, as for most of the test cases the WFA can find a better result than the other two. This is mainly because of the application of a local optimizer in the WFA, which exploits QoS-related information, while the two others don't have this ability. In addition, for most of the test cases, the WFA can find a feasible solution even though the constraint density is high.

The advantages of the proposed approach are as follows. In this research, an architecture is designed for dynamic and clear combination of web services.

This architecture helps the users to select a combination of services with respect to their qualitative parameters. Another innovation is that, a user can request for a dynamic combination of services. Also, using real dataset is a difference rather than other researches using generated data.

Conclusion and Future Work

In this research , QoS-Based web service selection and composition with the WFA meta-heuristic approachwere presented to find an optimal solution in a suitable time but the most important goal that should be paid attention to in service composition was the scalability of presented approach, because the weak scalability, when the problem size is increasing, makes the approach unusable. This goal was successfully provided by WFA approach. All test cases on previous studies were chosen in a way to observe the execution time of WFA approach with the increase in number of abstract and concrete web services and we studied the performance and scalability of the presented approach. The obtained results of simulation show good performance and scalability of approaches in large scale of tested cases.

In Section 5 the general architecture for automated web service selection and composition was presented. In this research only implementation and evaluation of one part of the architecture, which was service composition approach is studied, but other components of the architecture were described only briefly. Therefore, the next step of this research will consider the implementation of various components of the proposed architecture. Another future work is that one can increase the performance of WFA by combining it with another Meta heuristic approach such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO).

References

A. Tao and J. Yang, (2008),“Towards policy driven context aware differentiated services design and development”, Enterprise Information Systems 2(4), pp 367-384.

C. M. MacKenzie, "Reference model for service oriented architecture," in Public Review Draft 2, ed, 2006(accessed 2008-02-14).

C. Ma and Y. He, "An Approach for Visualization and Formalization of Web Service Composition," in International Conference on Web Information Systems and Mining, 2009. WISM 2009. , Shanghai, 2009, pp. 342-346.

C. Svensson and L. Wallen, "SOA and M & A-Relationships between Service Oriented Architectures (SOA) and Mergers and Acquisitions (M & A)," Department of Informatics, Lund University, Lund, Sweden, 2006.

C. Zhang, "Adaptive Genetic Algorithm for QoS-aware Service Selection," in Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on, Biopolis, 2011, pp. 273-278.

D. Paulraj, S. Swamynathan and M. Madhaiyan, (2012), “Process model-based atomic service discovery and composition of composite semantic web services using web ontology language for services (OWL-S)”, Enterprise Information Systems 6(4), pp 445-471.

E. Al-Masri and Q. H. Mahmoud, "Discovering the best web service," in Proceedings of the 16th international conference on World Wide Web, 2007, pp. 1257-1258.

E. Al-Masri and Q. H. Mahmoud, "Qos-based discovery and ranking of web services," in Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on, 2007, pp. 529-534.

E. Goncalves da Silva, et al., "Supporting dynamic service composition at runtime based on end-user requirements," in 1st International Workshop on User-generated Services, Stockholm, Sweden, 2009.

F. Rosenberg, A. Michlmayr and S. Dustdar, (2008), “Top-down business process development and execution using quality of service aspects”, Enterprise Information Systems 2(4), pp 459-475.

G. Zou, et al., "An agent-based web service selection and ranking framework with QoS," in Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on, Beijing, 2009, pp. 37-42.

H. Wang, "QoS-Based Web Services Selection," in e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on, Hong Kong, 2007, pp. 631-637.

J. M. Ko, et al., "Quality-of-service oriented web service composition algorithm and planning architecture," Journal of Systems and Software, vol. 81, pp. 2079-2090, 2008.

J. Wang and Y. Hou, "Optimal Web Service Selection based on Multi-Objective Genetic Algorithm," in

Computational Intelligence and Design, 2008. ISCID '08. International Symposium on, Wuhan, 2008, pp. 553-556.

L. Sha, "A QoS based web service selection model," in Information Technology and Applications, 2009. IFITA '09. International Forum on, Chengdu, 2009, pp. 353-356.

N. Griffiths and K. M. Chao, Agent-based service-oriented computing: Springer-Verlag New York Inc, 2010.

N. Josuttis, "SOA in practice-Art of Distributed System Design. 2007," ed: O'Reilly & Assoc, 2007.

S. Chaari, et al., "Enhancing web service selection by QoS-based ontology and WS-policy," in SAC '08 Proceedings of the 2008 ACM symposium on Applied computing, New York, NY, USA, 2008, pp. 2426-2431.

S. S. Yau and Y. Yin, "QoS-Based Service Ranking and Selection for Service-Based Systems," in Services Computing (SCC), 2011 IEEE International Conference on, Washington, DC, 2011, pp. 56-63.

S. Topouzidou, "SODIUM, service-oriented development in a unified framework," Final report ISTFP6-004559. http://www. atc. gr/sodium, 2007.

Wu, T. H., Chung, S. H., & Chang, C. C. (2010). A water flow-like algorithm for manufacturing cell formation problems. European Journal of Operational Research, 205(2), 346-360.

Y. Y. Fanjiang, et al., "Genetic algorithm for QoS-aware dynamic web services composition " in Machine Learning and Cybernetics (ICMLC), 2010 International Conference on, Qingdao, 2010.

Yang, F. C., and Wang, Y. P. (2007). Water flow-like algorithm for object grouping problems. Journal of the Chinese Institute of Industrial Engineers, 24(6), 475-488.

Z. Liang, et al., "A Hybrid Approach for the Multi-constraint Web Service Selection Problem in Web Service Composition⋆," 2012.